VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Metody shlukování ve zpracování obrazu

Clustering Methods in Image Processing

2010

Radomír Hladík

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava*.

V Ostravě 7. ledna 2010 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. ledna 2010 .....

Chtěl bych tímto poděkovat všem, kteří mi pomohli vypracovat tuto práci, zejména Ing. Janu Gaurovi.

Abstrakt

Cílem této práce bylo seznámit se s digitálním zpracováním obrazu s důrazem na metody shlukování. Zařadil jsem shlukovací metody do většího celku *segmentace* a rozepsal k čemu slouží, jak ji dělíme a které základní segmentační metody známe. V další části jsem se již dostal přímo k shlukovacím metodám. Ty jsem rozčlenil, podrobněji popsal jejich funkci a princip. Pro praktickou ukázku jsem zvolil shlukovací metodu mean-shift. Implementace, byla provedena pomocí knihovny OpenCV v programovacím jazyce C++. Následovala řada testů, které nám poskytly dostatek informací o funkčnosti algoritmu a jeho vlastnostech. Mean-shift dobře segmentuje i složitější obrazce (nemá problém s podlouhlými shluky) a výhodou je také, že není potřeba zadávat konečný počet shluků.

Klíčová slova: Digitální zpracování obrazu, segmentace, shluk, metody shlukování, OpenCV, mean-shift

Abstract

This bachelor thesis is focused on informing about digital image processing with lay emphasis on clustering methods. I inserted clustering methods to larger complex *segmentation* and specify how it serves, how we can divide it and which basic segmentation methods are known. I got down to clustering methods in another part. I divided them also described their functions and principles. I choose clustering method mean-shift for example. Implementation was made by support of library OpenCV in programming language C++. It followed set of tests which gave us enough information about algorithm functionality and his properties. Mean-shift segments even more difficult diagrams (doesn´t has any problem with long clusters) also there is one more advantage to set final quantity of clusters it is not necessary.

Keywords: Digital image processing, segmentation, custer, clustering methods, OpenCV, mean-shift

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
BCC	–	Borland C++ Compiler
GCC	–	GNU Compiler Collection
HW	–	Hardware
ICC	–	Intel C++ Compiler
IPP	–	Integrated Performance Primitives (knihovna spol. Intel)
MS	–	Mean-shift
MSVC6	–	Microsoft Visual C++ 6.0
OpenCV	–	Open Source Computer Vision
OS	–	Operační systém
PPC	–	PowerPC

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Dnešní moderní technika již celkem často využívá počítače pro automatickou analýzu dat z okolí. Můžeme se s ní setkat ve městech (měření hustoty provozu), při předpovědi počasí (analýza snímků z družice), bezpečnosti veřejných objektů (inteligentní kamery rozpoznávající osoby) apod. Většina počítačů a zařízení zpracovávajících informace z obrazu, se řídí podobným postupem při analýze obrazu: pořízení snímku z reálného světa (kamerou či fotoaparátem), následná digitalizace snímku (pokud je pořízen analogovým přístrojem), úprava snímku pro další zpracování, analýza snímku (rozpoznání objektů) a závěrečné vyhodnocení dat získaných z obrazu (počet osob, objektů apod.). V této práci se budu zabývat jednou částí z celého tohoto procesu, a to: metody shlukování při digitálním zpracování obrazu. Tato technika úpravy obrazu spadá do většího celku, který se nazývá segmentace obrazu a je jednou z nejdůležitějších částí celého zpracování. Proto nejprve popíši, co je to segmentace obrazu, pak vysvětlím souvislost s metodami shlukování a jaké metody se používají. Z těchto metod si vyberu jednu, tu důkladně popíšu a následně provedu implementaci pomocí knihovny OpenCV. Pro názornou ukázku metody shlukování jsem si vybral algoritmus mean-shift.

2 Segmentace obrazu

Obecně bychom mohli segmentaci popsat, jako rozdělení jednotlivých částí objektu do skupin s podobnými vlastnostmi. V digitálním zpracování obrazu se segmentace používá k rozčlenění analyzovaného obrazu do oblastí, které mají úzkou souvislost s objekty reálného světa zachycenými na obraze. Výsledkem segmentace na obraze by měly být oblasti, které se nepřekrývají a reprezentují jednotlivé objekty ze zachyceného obrazu, pak jde o kompletní segmentaci. Další možností je, že výsledné oblasti nebudou odpovídat těmto objektům, pak tuto segmentaci nazýváme jako částečnou [1] [2].

Kompletní segmentace obecně využívá vyšší úroveň zpracování, která je založena na znalostech řešeného problému. Také ji můžeme získat při zpracování jednodušších obrazů, které mají dobře patrné rozdíly mezi pozadím a popředím (např. texty). Proto z obrazů pořízených v reálném světě většinou dostaneme částečnou segmentaci, která je založena na principu homogenity vlastností obrazu (např. jas, barva). Dosažení kompletní segmentace u složitějších obrazců není zpravidla možné, a proto postačí získat částečnou segmentaci. Ta nám výrazně zredukuje objem dat, který bude dále zpřesněn operacemi na vyšší úrovni. Segmentaci může negativně ovlivnit šum v zachyceném obraze, nepředvídané osvícení objektu, počasí. Pokud však budou tato rušení předvídána, lze je pomocí speciálních algoritmů potlačit [1] [2].

V praxi se se segmentací můžeme setkat velmi často, proto zde uvedu pár příkladů:

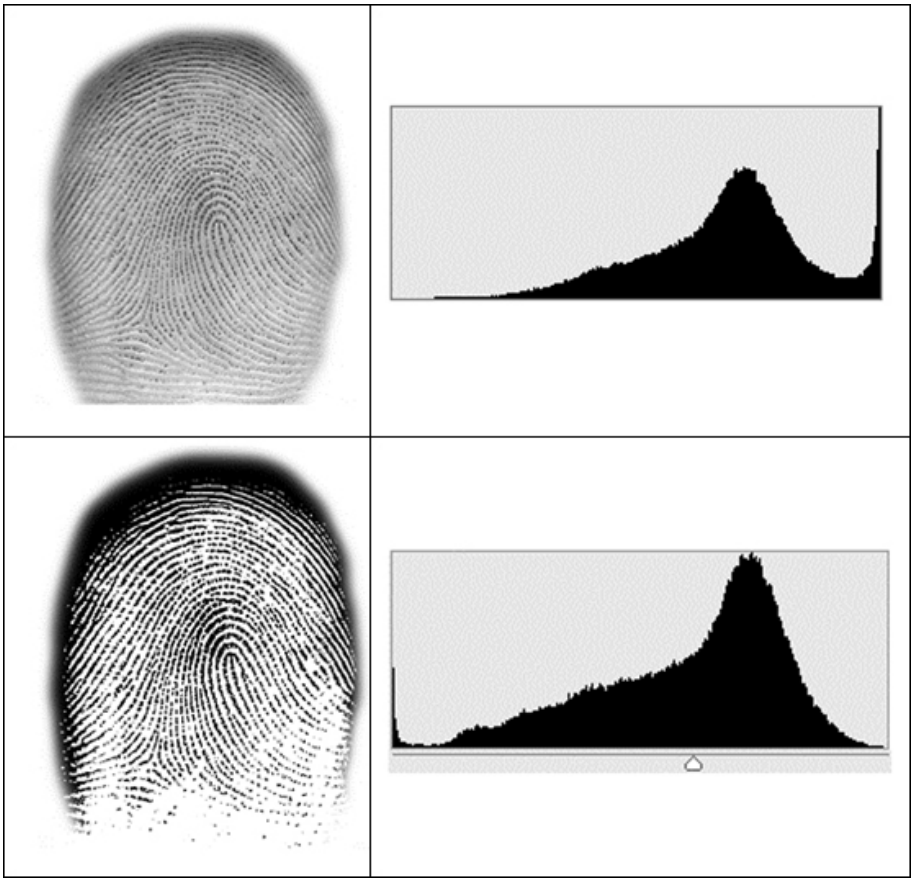
- zpracování lékařských obrazových dat (nalezení nádoru, měření objemu tkáně, . . .)
- nalezení objektů v satelitních snímcích (cesty, lesy, pole, . . .)
- rozpoznávání otisků prstů, obličejů
- řídící systémy v dopravě
- počítačové vidění

Pro segmentaci je užíváno mnoho metod a často musí být použity v souvislosti se znalostmi o řešeném problému, abychom dosáhli požadovaného výsledku. Známe tyto segmentační metody: založené na růstu regionů, shlukování, prahování, detekce hran nebo segmentace rozvodí. Toto je jen část z používaných segmentačních metod, ale k porozumění principů segmentace postačí [2].

2.1 Prahování

Prahování je jednou z nejjednodušších metod segmentace. Je založená na rozdělení obrazu s více úrovněmi jasu (šedotónového obrazu) na obraz, kde jsou pouze dvě jasové úrovně (černá, bílá). Pro toto rozdělení se využívá histogram[1], pomocí nějž si zvolíme takový práh (threshold), aby pixely s menší hodnotou jasu než zvolený páh byly brány, jako pozadí (jas 0) a pixely s větší hodnotou, jako popředí (jas 255). Samozřejmě, že si můžeme

[1] Je to graf, kde na ose x je vynesen odstín (0 – 255) a na svislé ose y je znázorněn počet pixelů v dané úrovni jasu.



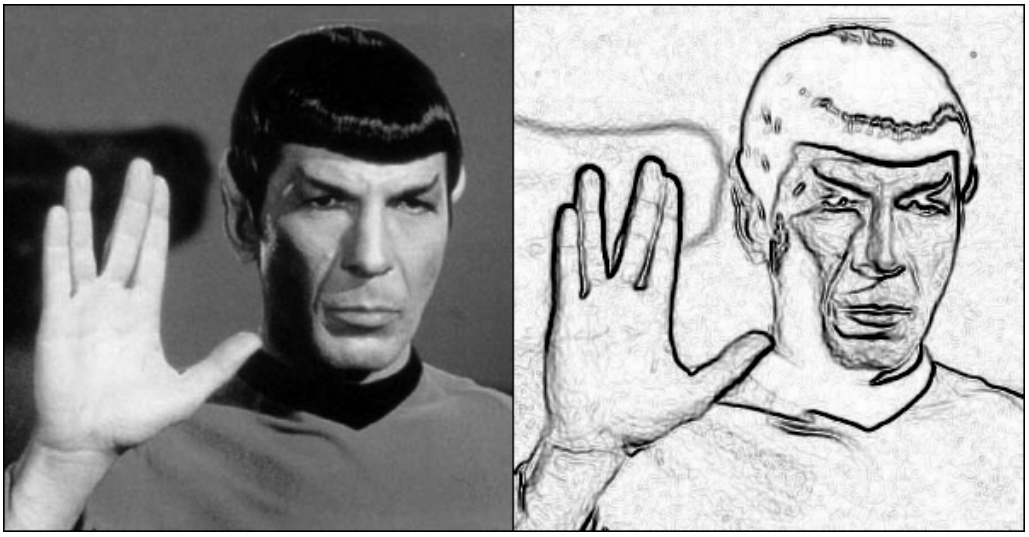
Obrázek 1: Prahování

navolit, aby rozdělení bylo prováděno obráceně (záleží na tom co potřebujeme z obrazu vysegmentovat). Na obrázku 1 můžeme vidět vstupní obraz (šedotónový obraz) před prahováním a výstup po prahování. Také je dobré si povšimnout histogramu vpravo, kde je zobrazena navolená hodnota prahu [3] [4].

Někdy však není možné určit globální práh pro celý obraz, protože obraz osahuje více různě jasných oblastí. Na takovéto obrazy se však dá použít adaptivní prahování. Což je metoda, která mění velikost prahové hodnoty dynamicky nad obrázkem, který zpracovává. To znamená, že pro každý pixel v obraze se počítá nová prahová hodnota. Tato lépe propracovaná verze prahování umožňuje přizpůsobit se stavu osvětlení v obraze, tj. velkým změnám intenzity osvětlení nebo stínům [3] [4].

2.2 Detekce hran

Tento druh segmentace slouží k nalezení obrysů (nebo-li hran) objektů na obraze. Hranou rozumíme řadu bodů, jejichž jas prudce kolísá. Také ji můžeme popsat jako vlastnost



Obrázek 2: Detekce hran

obrazového bodu započteného jako funkci obrazu v okolí tohoto bodu. Hrana je pak vyjádřena svou velikostí a směrem [3] [4].

Pro nalezení těchto hran používáme takzvané detektory hran. Ty si můžeme rozdělit na dvě základní skupiny, podle způsobu detekce: detektory využívající první derivaci a druhou derivaci. Pokud použijeme první derivaci, porovnáváme výsledný hranový gradient s prahem, který nám určí zda jde o hranu či nikoli. Na proti tomu, druhá derivace posuzuje, zda je prostorová změna v polaritě druhé derivace dostatečně velká. Zástupci nejznámějších detektorů hran jsou: Robertsův, Prewittové, Sobelův, Kirschův. Po jejich aplikaci dostaneme obraz se zdůrazněnými hranami, ale stále zde máme nepřesnosti způsobené šumem nebo lokálními nehomogenními oblastmi. Proto tyto obrazy musíme dále zpracovávat (např. metodou prahování). Na obrázku 2 můžeme vidět, jak by taková detekce hran mohla vypadat [3] [4].

2.3 Aktivní kontura (active contour, snake)

Další segmentační metodou je *aktivní kontura*. Patří k pokročilejším metodám segmentace a vyžaduje, aby ve vstupním obraze byla křivka, která zhruba ohraničuje objekt, který má být později přesně ohraničen. Pro přesné ohraničení objektu tuto vstupní konturu deformují tzv. vnitřní vlivy obrazových a vnějších sil. Vnitřní síly kontrolují hladkost průběhu, obrazové síly řídí tvarování kontury směrem k hraně objektu a vnější síly jsou dány volbou počáteční kontury (tvar, umístění, . . .) [23].

Ačkoli tato metoda dosahuje dobrých výsledů a má více výhod něž většina ”základních” metod, dochází i zde k nechtěným anomáliím, pokud pracujeme se složitějším obrazcem. Vytváří falešné kontury a nežádané smyčky. Proto byla později vytvořena modifikovaná verze, odstraňující některé nedostatky. Na obrázku 3 můžeme vidět, jakých výsledků dosahuje aktivní kontura standartní a modifikovaná [23].

Obrázek 3: Vstupní křivka aktivní kontury, základní segmentace, rozšířená metoda segmentace [23]

2.4 Regionální metody

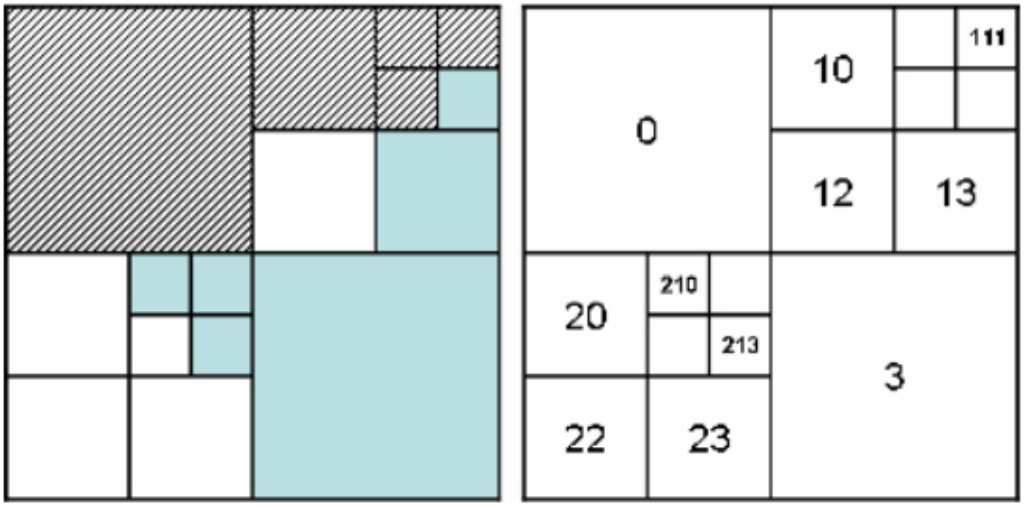
Oproti předchozí metodě se tentokrát nehledají hrany, ale celé oblasti. Díky tomuto rozdílu není tato segmentace tak náchylná na šum, což byl problém u detekce hran. Segmentace se provádí hledáním bodů s podobnými vlastnostmi (jas, barva, textura, tvar a pod.).

Jedna z těchto metod je metoda region growing (růst regionů). Funguje tak, že nejprve se v obraze rovnoměrně nebo náhodně rozmístí tzv. semínka (počáteční pixely), ze kterých se vychází. Každé semínko porovná okolní pixely a pokud jsou si podobné, pak dojde ke sloučení do jedné oblasti. Tyto oblasti se dále iterativně rozrůstají a rozšiřují se. Nevýhodou je, že pokud dojde k jinému rozmístění nebo zvolení jiného počtu semínek, nemusí být výsledná segmentace stejná [3] [6].

Další metodou je split and merge, která vznikla ze starších metod region merging a region splitting. Obraz je v této metodě postupně dělen na menší a menší části (podle předem dané struktury) a zároveň se sousední oblasti spojují, pokud splňují kritéria homogenity. Než začneme obraz segmentovat, je nutné si zvolit způsob dělení oblastí. Nejčastěji se používá quadtree . Je to stromová struktura, v níž se každá oblast při přechodu na nižší úroveň dělí na čtyři čtvercové oblasti (viz. obrázek 4). Další vlastnost, kterou musíme nadefinovat, je kritérium homogenity. Tato vlastnost je velmi důležitá a ovlivňuje chování celého algoritmu. Při definování kritérií homogenity bychom měli brát v potaz, jaké objekty se na obraze budou vyskytovat a jak se budou projevovat. Musím však upozornit, že kritéria homogenity pro rozdělování oblastí nemusí být stejná pro jejich slučování [4] [6].

2.5 Záplava (watershed)

Algoritmus záplava je segmentační metoda matematické morfologie. Hlavní rozdíl oproti ostatním segmentačním metodám je, že se na obrazovou funkci nahlíží, jako na reliéf krajiny, která je zaplavována vodou. Maximální počet segmentů je roven počtu lokálních minim (nížin). V těchto nížinách vyvěrá voda zaplavující okolí. Pokud dojde k slití vody ze dvou různých pramenů, postaví se na tomto místě hráz. Algoritmus končí až je celá



Obrázek 4: Split and merge [5]

krajina pod vodou a výsledkem segmentace je síť hrází. Na obrázku 5 můžeme vidět, jak tento algoritmus postupuje [3] [4].

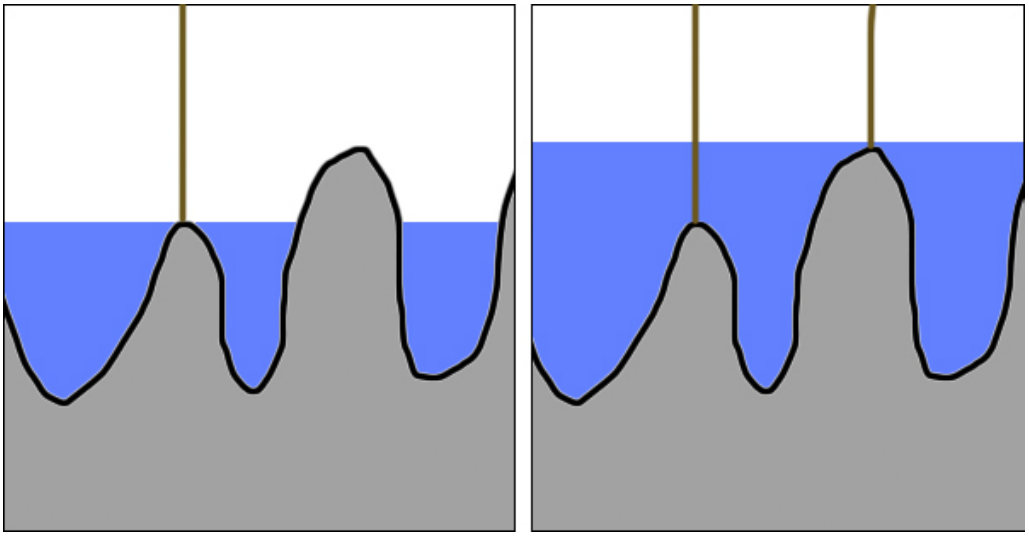
2.6 Active Appearance Models

Tuto metodu bychom mohli zařadit k tzv. *"znalostním"*. Je to metoda segmentace, která potřebuje pro svůj běh trénovací data, která jsou reprezentována šablonou nebo modelem objektů, které se později budou segmentovat. Tato trénovací data jsou vytvářena z ručně segmentovaných obrazů, kde postupně zadáváme hraniční body. Množinu šablon nebo modelů je třeba ručně anotovat. Při samotné segmentaci je pak porovnáván tvar objektů a intenzita pixelů [23].

Výhodou je velmi rychlé zpracování, pokud se jedná o objekty podobné trénovacím datům. Pokud se však bude jednat o jiné objekty, dostaneme většinou velmi špatné výsledky. Další nevýhodou je právě vytváření trénovacích dat, což nám může zabrat mnoho času. Protože se při segmentaci zaměřujeme na objekty stejného charakteru, zdálo by se, že využití bude velmi omezené. Opak je však pravdou. Hojně se využívá v medicínských aplikacích, při rozpoznávání tvarů v obraze (obličeje, výrazů obličeje, předmětů a pod), kde nám nevadí zadávání trénovacích dat, ale je podstatná rychlost zpracování [23].

2.7 Metody shlukování

Posledním, a v této práci nejdůležitějšími metodami segmentace jsou shlukovací metody. Poněvadž se těmto metodám bude věnovat celá následující kapitola, nastíním zde jen obecný popis. Metody shlukování pracují na principu třídění jednotlivých prvků do skupin (shluků) tak, aby jednotlivé prvky ze stejné skupiny si byly více podobné než prvky ze skupin jiných. Vytváření těchto skupin nazýváme shlukování. Kvalita shlukování je



Obrázek 5: Záplava (watershed)

přímo závislá na schopnosti algoritmu, porovnat odlišnosti mezi jednotlivými prvky (objekty). Schopností shlukování se využívá ve zpracování obrazu, datové analýze nebo při dolování dat(*data mining*).

3 Metody shlukování

Shlukování nebo-li shluková analýza není pouze metoda segmentace obrazu. Řadí se do skupiny metod učení bez učitele a jejím cílem je rozdělit danou množinu objektů do takových podmnožin, aby objekty patřící do téže podmnožiny měly k sobě "blíže" (byly si podobnější, méně nepodobné), než objekty z jiných podmnožin. Aby bylo zcela jasno co je shluková analýza (dále již jen shlukování), uvedu zde několik příkladů charakteristiky shlukování [7]:

- **R.C.Tryon (1939):** *"Shluková analýza je obecný logický postup formulovaný jako procedura, pomocí níž seskupujeme objektivně jedince do skupin na základě jejich podobností a rozdílností."*
- **R.E.Bonner (1964):** *"Je dána množina objektů, z nichž je každý definován pomocí množiny znaků s ním souvisejících. Tato množina znaků je pro každý objekt stejná. Máme nalézt shluky objektů (podmnožiny původní množiny objektů) tak, aby si členové shluku byli vzájemně podobní, ale nebyli si příliš podobní s objekty mimo tento shluk."*
- **M.R.Anderberg (1975):** *"Tento problém je obvykle charakterizován jako hledání přirozených skupin. Konkrétněji jde o třídění pozorování do skupin tak, aby stupeň přirozené asociace členů téže skupiny byl vyšší a členů různých skupin nižší."*

Bohužel, v literatuře se vyskytuje velké množství shlukovacích metod, takže je problém, je nějak rozumně utřídit. Proto jsem zvolil náhled na shlukovací metody ne podle použitých matematických prostředků, ale podle toho, jakého výsledku dosáhneme. Vzniknou nám tedy dvě základní skupiny: *hierarchické* a *nehierarchické metody*.

3.1 Objekty a znaky

Pokud budeme třídit objekty podle předem zadaných kritérií, musíme před samotným tříděním objekty zvážit, změřit, zjistit barvu, určit všechny jejich význačné znaky, podobnosti a odlišnosti. Objektem rozumíme útvar popsaný pomocí konečného počtu charakteristik - znaků. To znamená, že objekt je popsán *n-ticí* stavů předem stanovených *n znaků*. Stavy většinou označujeme číselně, a tudíž objekty shlukování jsou n rozměrné vektory čísel. [7] [8]

1. Kvalitativní znaky
    - Hodnoty znaků náleží do konečné množiny možných stavů, kterým můžeme přiřadit číselný kód. Tyto znaky můžeme rozdělit na *nominální* např. tvar (1 - kruh, 2 - čtverec, 3 - kvádr), a *ordinální*, což znamená, že se tyto hodnoty mohou seřadit, např. věk (1 - mladý, 2 - střední, 3 - starý).
    - Mezi kvantitativní znaky se také řadí *binární* (dichotomické) znaky. Hodnota patří do dvouprvkové množiny např. pohlaví (muž / žena) nebo mít auto (true / false) [7] [8].

2. Kvantitativní znaky

   - Jsou to takové znaky, jejichž hodnoty vyjadřují množství. Bývají pak popsány číslem patřícím do konečné, spočetné či nespočetné množiny. Čísla mohou být reálná nebo celá, příkladem kvantitativního znaku nám může být délka, teplota.

3.2 Standardizace dat

Velmi často se stává, že některé hodnoty znaků objektů jsou řádově vyšší/nižší než jiné znaky. Takovéto nepoměry jednotlivých hodnot znaků jsou většinou nechtěné, protože pak může dojít k malému ovlivnění shlukování těmito hodnotami. Tyto velké rozdíly může způsobit použití jiných jednotek nebo zaznamenání naprosto odlišných vlastností objektu (barva 1 - 5 nebo věk 1 - 110). Proto je někdy vhodné data upravit tak, aby byla souměřitelná. Provádí se to například standardizací dat [7] [8].

Představme si, že chceme standardizovat n objektů s m znaky. Zapíšeme si tedy tyto informace do matice o n řádcích a m sloupcích $X = (x_{ij})$, kde $x_{ij} \in R$. Standardizace se provádí tak, že si vypočteme pro každý sloupec $j = 1, \ldots, m$ průměrnou hodnotu j-tého znaku [7] [8]

$$x_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$$

dále pak, směrodatnou odchylku j-tého znaku

$$s_j = \left[\frac{1}{n} \sum_{i=1}^{n} (x_{ij} - x_j) \right]^{1/2}$$

Po té hodnoty x_{ij} přepočítáme na tzv. standardizované hodnoty

$$z_{ij} = \frac{x_{ij} - x_j}{s_j}$$

3.3 Podobnost a nepodobnost objektů

Ve shlukování se často řeší problematika podobnosti objektů a jejich číselného vyjádření. Číselným vyjádřením máme na mysli to, že pokud máme dva objekty, jejich vzájemná podobnost bude vyjádřena číselnou hodnotou. Dobrým způsobem, jak určit tuto hodnotu, je zavedení funkce podobnosti $\Pi : O \times O \to R_0^+$, která přiřadí dvěma objektům reálné nezáporné číslo $\Pi(o_h, o_s)$, pro něž musí platit [7] [8]:

$$\Pi(o_h, o_s) \geq 0$$

$$\Pi(o_h, o_s) = \Pi(o_s, o_h)$$

Pokud tyto požadavky dodržíme, bude platit, že čím si budou dva objekty podobnější, tím bude hodnota $\Pi(o_h, o_s)$ větší. Z toho by nám vyplývalo, že pokud dva objekty budou totožné, bude jim přiřazena maximální číselná hodnota. U shlukovacích metod však bývá vhodnější, aby tato hodnota byla nulová, proto si zavedeme pojem nepodobnost objektů $d\ O \times O \rightarrow R_0^+$, pro které platí [7] [8]:

$$d(o_h, o_s) \geq 0 \Leftrightarrow o_h = o_s$$

$$d(o_h, o_s) \geq 0$$

$$d(o_h, o_s) = d(o_s, o_h)$$

Z těchto vlastností vyplývá, je-li nepodobnost dvou objektů rovna nule, jsou tyto objekty totožné. A čím víc bude růst míra nepodobnosti, tím méně si budou tyto objekty podobné. V praxi se pro určení nepodobnosti d, používají různé metriky [2] definované v prostoru R^m. Nejčastěji se využívá Euklidovské metriky. Máme-li tedy objekty charakterizované p znaky, můžeme si je převést na body p-rozměrného euklidovského prostoru E_p. Mezi těmito body (h,s), pak můžeme určit euklidovskou vzdálenost takto [7] [8]:

$$d(o_h, o_s) = \left(\sum_{j=1}^{m} (x_{hj} - x_{sj}) \right)^{1/2}$$

Pokud však nemůžeme objekty popsat pomocí vektoru R^m (což jsou objekty s kvalitativními a binárními znaky), pak se využívá slovního popisu nebo zápisu do tabulek. Abychom mohli tyto zápisy vyjádřit i matematicky, zavádí se tzv. *koeficient asociace*, který se na podobnost dvou objektů dívá trochu jinak: $A(o_h, o_s) = s/m$, kde m je celkový počet znaků a s je počet shodných kladných znaků. *Koeficient asociace* lze vyjádřit i jako: $A(o_h, o_s) = (s + z)/m$, kde z je počet shodných záporných znaků. Platí zde $0 \leq A \leq 1$, na základě tohoto lze nepodobnost dvou objektů definovat tímto vztahem [7] [8]:

$$d_A(o_h, o_s) = 1 - A(o_h, o_s)$$

3.4 Shluk objektů

Pojem *shluk* se již v tomto textu objevil, ale prozatím nebyl řádně vysvětlen a formálně definován. Avšak, když už jsme si vysvětlili, co je to podobnost (nepodobnost), můžeme se o definici pokusit. Záměrně říkám *pokusit*, protože *shluk* je velmi obecný a není

[2] Metrika, nebo-li vzdálenost, vychází z pojmu metrický prostor. Což je matematická struktura, pomocí které lze formálně definovat pojem vzdálenosti. Může být matoucí, proč se bavíme o vzdálenosti, když právě řešíme podobnost. Avšak z matematického hlediska, můžeme podobnost vyjádřit, jako vzdálenost. Tudíž, jsou li si dva objekty blízké, jsou si podobné.

možné nalézt jeho všezahrnující definici. Jednou z možností definování shluku je pomocí nepodobnosti objektů. *Shluk* tedy je taková podmnožina **A** objektů **O**, pro niž platí $(o_i, o_j, o_k \in A; o_l \notin A)$ [7] :

$$\max d\,(o_i, o_j) < \min d\,(o_k, o_l)$$

Pokud se bavíme o *shluku* a shlukovacích metodách, dříve nebo později narazíme na problém, jak určit, do kolika shluků objekty rozdělit. U některých shlukovacích metod se počet shluků zadává před spuštěním algoritmu, u jiných se určí během provádění. Také je důležité zmínit, že každý objekt shlukování může patřit pouze do jednoho shluku a každý shluk obsahuje minimálně jeden objekt. Rozdělení objektů do shluků bychom si mohli zapsat pomocí matice $U = (i_{ij})_{c,n}$, pro kterou platí: [7]

$$u_{ij} = u_i\,(o_j) = 1\text{ , jestliže } o_j \in S_i a u_{ij} = u_i\,(o_j) = 0\text{ , pak } o_j \notin S_i$$

Také je důležité, aby platilo:

$$0 < \sum_{j=1}^{n} u_i < n \quad \text{a} \quad \sum_{i=1}^{c} u_j = 1$$

Pokud matice U splňuje výše uvedené požadavky, nazveme ji *c-rozkladem* a množinu všech *c-rozkladů* M_c, kde platí:

$$M_c = \left\{ U \in V_{cn}, u_{ij} \in \{0,1\} \forall_{i,j}; \sum_{i=1}^{c} u_{ij} = 1 \forall_j; 0 < \sum_{j=1}^{n} u_{ij} < n \forall_i \right\}$$

V_{cn} zde znamená vektorový prostor dimenze cn.

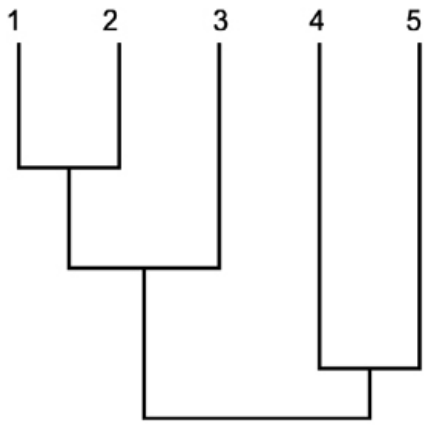
Některé shlukovací metody také potřebují porovnávat celé shluky (jde třeba o vyčíslení rozdílu shluků mezi sebou). Proto je pro shluky definovaná *nepodobnost* shluků d_s, která formálně vychází z nepodobnosti objektů. Pro shluky se také používá koeficient nepodobnosti shluků. Nejčastější jsou tyto: *metoda nejbližšího souseda, metoda nejvzdálenějšího souseda, centroidní metoda, metoda průměrné nepodobnosti*.

3.5 Hierarchické shlukování

Výsledkem této metody je binární strom (také nazývaný *dendrogram*), znázorňující postupné shlukování objektů a jejich hierarchické uspořádání. Na jedné straně tohoto stromu je každným objektem tvořen jednoprvkový shluk a na druhé straně je jeden shluk obsahující všechny objekty. Počet shluků si pak můžeme buď vybrat (podle potřeby) a nebo zvolíme metodu, která využívá shlukovacích hladiny (viz. níže). Zde se volí takový rozklad, pro který je rozdíl od následující hladiny výrazně větší než ostatní rozdíly. [8] [10] [11]

Hierarchické metody se dále dělí podle rozkladu na binární strom takto:

- aglomerativní shlukování
- divizní shlukování



Obrázek 6: Dendrogram

3.5.1 Aglomerativní shlukování

Hned na počátku je vytvořeno *n* jednoprvkových množin, obsahujících právě jeden objekt z množiny objektů O. Tuto úpravu nazýváme jako nultý rozklad Ω_0.Pro další postup je třeba definovat kritéria, na jejichž základě se bude porovnávat nepodobnost (podobnost) objektů. V každém následujícím kroku se pak provede shluknutí dvou objektů do jednoho, čímž se bude celkový počet objektů stále snižovat, až nám zbude pouze jeden. Celý algoritmus by se dal shrnout do těchto tří kroků: [8] [10] [11]

- Provede se již zmíněný nultý rozklad Ω_0 tak, aby každý objekt z množiny objektů O, tvořil právě jeden jednoprvkový shluk $A_{0,i} = \{o_i\}$. Zde platí: $\Omega_0 = \{A_{0,1}, A_{0,2}, \ldots, A_{0,n}\}$. Následně se každému shluku přiřadí číslo $h\left(A_{0,i}\right) = \mu_0 = 0$, což je shlukovací hladina.

- V *i-tém* kroku $(0 < i \leq n-2)$ se vytvoří rozklad $\Omega_i = \{A_{i,1}, A_{i,2}, \ldots, A_{i,n-i}\}$, tak aby vždy ze všech shluků byly vybrány dva nejvíce si podobné (nejméně si nepodobné). Tyto podobné shluky ($d\left(A_{i,u}, A_{i,v}\right) = \mu_i$) jsou pak sloučeny do jednoho ($A_{i,u} \cup A_{i,v} = A_{i,l}$) a všechny ostatní ponechány bez změny.

- V posledním kroku dojde ke sloučení všech shluků do jednoho, kde bude platit, že konečný rozklad obsahuje všechny objekty z množiny $O : \Omega_{n-1} = \{A_{n-1,l}\} = O$. Pomocí informací rozkladu $\Omega_0, \Omega_1, \ldots, \Omega_{n-1}$ a shlukovacích vrstvách $\mu_0, \mu_1, \ldots, \mu_{n-1}$, můžeme sestavit podobnostní strom (dendrogram) viz. obrázek 6. Z tohoto příkladu je patrné:

  $\mu_0 = \{(1), (2), (3), (4), (5)\}$;

  $\mu_1 = \{(1,2), (3), (4), (5)\}$;

  $\mu_2 = \{(1,2,3), (4), (5)\}$;

  $\mu_3 = \{(1,2,3), (4,5)\}$;

$\mu_4 = \{(1, 2, 3, 4, 5)\}$

3.5.2 Divizní shlukování

U divizního shlukování se postupuje přesně opačným způsobem než u aglomerativního shlukování. Na počátku máme jeden shluk obsahující všechny objekty z množiny O a v každém následujícím kroku rozdělíme všechny existující shluky na menší shluky (nejčastěji dva). Algoritmus rozkládá tyto shluky tak dlouho, dokud v každém shluku nezůstane pouze jeden objekt. Bohužel tato metoda je výpočetně náročná. K rozložení daného shluku s n objekty na dva shluky, podle námi stanoveného kritéria podobnosti (nepodobnosti), je třeba vyzkoušet $2^{n-1}-1$ možností. Kvůli tomuto se tyto metody nehodí pro zpracování většího objemu dat. [8] [10]

4 Nehierarchické shlukování

U nehierarchických metod shlukování se snažíme dosáhnout takového rozkladu objektů O na shluky $S = \{S_1, \ldots, S_n\}$, aby vyhovoval námi zadaným kvalitativním požadavkům. To znamená, že algoritmus se bude provádět tak dlouho, dokud nedojde do extrémních hodnot u vlastností shluků tvořících rozklad. Naše kvalitativní požadavky by měly obsahovat některé z těchto vlastností: podobnost objektů ve shluku, míru separace shluků, rovnoměrné rozložení do různých shluků. [8] [10] [11]

Tyto algoritmy se při svém řešení zpravidla zaměřují na dva problémy. Na nalezení nejvhodnějšího počtu shluků k pro každou množinu objektů O a nalezení optimálního rozkladu množiny objektů O na k shluků $S = \{S_1, \ldots, S_n\}$.

Nehierarchické metody by se ještě dále mohly dělit podle mnoha kritérií, ale v téměř každém zdroji se toto dělení různí. Proto se již dělením dále nebudu zabývat a popíšu jen typické metody z této skupiny. [7] [8]

4.1 K - means

K-means nebo-li *metoda nejbližších středů*. Patří do skupiny nehierarchických metod, která má hned na počátku zadaný počet tříd (shluků), kterých chceme dosáhnout. Přidělování do jednotlivých tříd probíhá na základě Euklidovské vzdálenosti objektu od středu shluků. Během provádění algoritmu dochází k přeskupování shluků. Tím rozumíme, že objekty mohou přecházet z jedné tříd do druhé. [8] [9] [11]

Algoritmus probíhá ve dvou základních krocích, kterým předchází počáteční vytvoření k shluků. Výběr počátečních objektů reprezentujících shluky by měl být pečlivý (i když v praxi se volí i náhodně), jelikož pro každý výběr bývá výsledek jiný. Výběrem počátečních objektů jsme si nejen vytvořili k shluků, ale také jsme jimi stanovili střed shluku pro každou třídu. Po inicializaci tříd se začnou iterativně opakovat následující dva kroky (ukázka postupu k-means viz. obrázek 7) [8] [9] [11]:

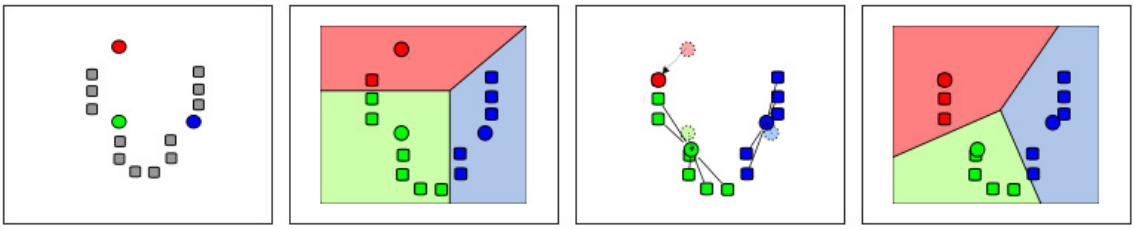
- Všechny objekty $x_i, i = \{1, \ldots, l\}$ jsou zařazeny do tříd určenými vektory $\mu_i, i = 1, \ldots, k$, podle minima euklidovské vzdálenosti. Tedy objekt x_i je zařazen do třídy y_i podle:

$$\arg_j \min \sum_{j=1}^{k} \sum_{x_i \in S_i} \|x_i - \mu_j\|^2$$

- Nyní se přepočítají vektory μ_j tak, aby odpovídaly střední hodnotě všech objektů x_i v každé třídě y_i. Výpočet nové hodnoty se provede podle následujícího vzorce, kde l_j je počet objektů x_i, které byly v předchozím kroku zařazeny do třídy s vektorem středu μ_j. [8] [9] [11]

$$\mu_j = \frac{1}{l_j} \sum_{i=1, y_i=j}^{l} (x_i)$$

Tyto dva kroky se opakují tak dlouho, dokud již nedojde k přesunu objektu z jedné třídy do druhé (končí v lokálním extrému). Toto řešení navržené McQueenem bylo později



Obrázek 7: Postup algoritmu k-means [12]

upravené Whishartem tak, že k ukončení dojde až po dvou po sobě jdoucích iteracích, během nichž nedošlo ke změně. K-means algoritmus je vhodný pro využití ve vektorové kvantizaci, protože nemá přehnané požadavky na výpočetní výkon a kvalita rozkladu je také dostačující. [8] [9] [11]

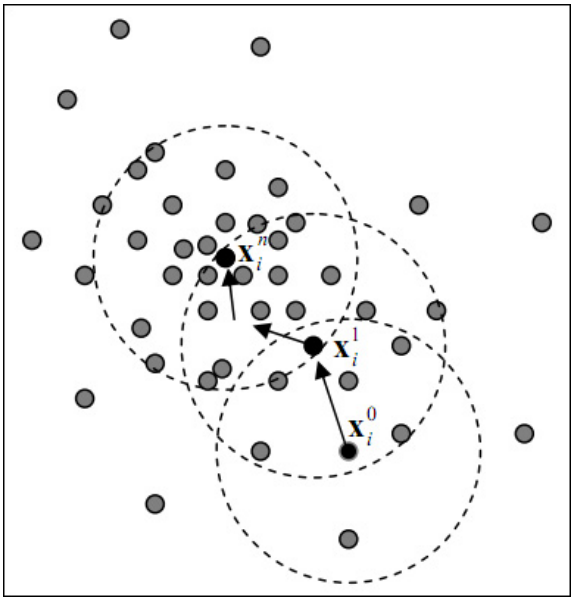
4.2 Mean-shift

MS je obecný algoritmus pro iterativní nalezení lokálního maxima hustoty vzorků. Spadá do kategorie tzv. algorytmů *učení bez učitele*, pro svůj běh potřebuje pouze dva parametry(čistě fyzikálního významu). Poprvé byl představen v roce 1975, Fukunagou a Hostetlerem. Později byl postupně upravován pro digitální zpracování obrazu, *low-level vision* včetně segmentace, adaptivní vyhlazování a sledování objektů. Využití toho algoritmu je opravdu široké, ale my se zaměříme pouze na segmentaci. [13] [14] [15]

V tomto algoritmu jsou body shlukovány na základě podobnosti jejich vzhledu a blízkosti jejich pozic. Z toho si můžeme vyvodit, že body budou patřit do stejného shluku, pokud jsou si podobné a jejich nejbližšší lokální maximum hustoty je stejné. Proto budeme hledat v okolí bodu lokální maximum hustoty. To se provádí tak, že z okolí bodu (což je *první parametr*- tvoří jakési okénko, viz. obrázek 8) si vypočteme místo s největší hustotou a přesuneme se na něj. Tento postup se provádí tak dlouho, dokud nedojdeme do lokálního maxima hustoty a tudíž se již nemáme kam přesunout. Celý algoritmus bychom si mohli rozdělit na dva kroky [13] [14] [15]:

- Z každého objektu z množiny objektů, spustíme means-shift a zapamatujeme si lokální maximum do nějž dokonvergoval.
- Jednotlivé shluky pak vytváříme z těch bodů, které dokonvergovaly do stejného maxima s určitou tolerancí (*druhý parametr*).

Díky způsobu segmentace není tato metoda omezena tvarem segmentu a zvládá i podlouhlé shluky. Proto se hodí například pro medicínské přístroje zpracovávající snímky tkání. Bohužel bez předchozího *"zjednodušení"* je algoritmus náročný na výpočetní výkon a čas. Podrobnější popis tohoto algoritmu popíši v části implementace.



Obrázek 8: Mean-shift posun [25]

4.3 Fuzzy c-means

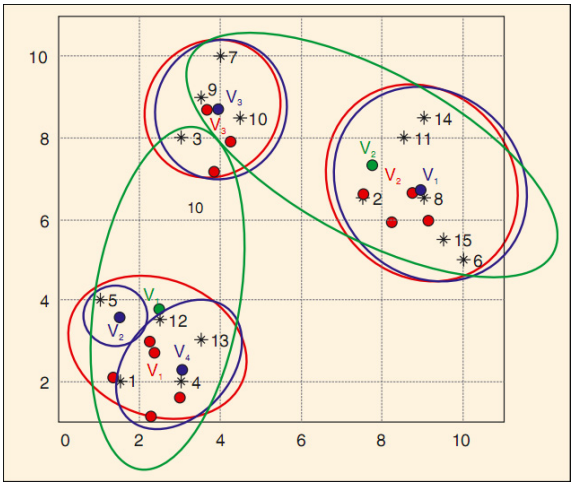
Prozatím jsme se bavili jen o metodách, které přesně přiřazovaly objekty do shluků. Jeden objekt tedy mohl patřit pouze do jednoho shluku. Stává se však, že toto rozdělení není dostačující a my potřebujeme zjistit míru příslušnosti k jednotlivým shlukům. Tedy objekty mohou patřit do nějaké třídy víc než do jiné. Příkladem nám třeba může být zařazení ptakopyska do tříd savci i ptáci. Pokud bychom dělení provedli jednoznačně, pak by byl savec a ne pták. Což však není úplně správné. Proto se zavedly tzv. fuzzy množiny, které již nevyužívají pouze dvě hodnoty $\{0, 1\}$, ale využívají hodnoty z intervalu $\langle 0, 1\rangle$. S využitím fuzzy množiny by ptakopysk mohl patřit do třídy savci s příslušností 0,8 a do třídy savci s 0,2 [16] [17].

I když použijeme fuzzy dělení mohou být na něj kladeny stejné požadavky na příslušnost, jako v předchozích případech. Objekt s příslušností 1 patří do shluku a každý shluk musí obsahovat víc než 0 a méně než n shluků. Tyto podmínky si můžeme vyjádřit takto:

$$\sum_{i=1}^{c} u_{ij} = 1 \text{ a } 0 < \sum_{j=1}^{n} u_{ij} < n$$

Představeným algoritmem z rodiny fuzzy bude c-means. Byl navržen roku 1973 panem Dunnem a později vylepšen Bezdekem 1981. Patří do skupiny jednodušších fuzzy algoritmů a pro jeho provedené je třeba znát předem počet shluků $(c > 2)$. Princip je velice podobný k-means, má i podobné problémy, ale je jich méně. Opět si algoritmus rozdělíme do bodů, pro lepší pochopení. [16] [17]

- Nejprve je třeba zvolit si počáteční středy shluků c_k



Obrázek 9: Příklad výsledku fuzzy shlukování [18]

- Poté začneme s výpočtem příslušnosti všech objektů x_n k shlukům S_k:

$$\alpha_{n,k} = \frac{\frac{1}{\|x_n - c_k\|^{\frac{2}{m-1}}}}{\sum_j \frac{1}{\|x_n - c_j\|^{\frac{2}{m-1}}}}$$

  kde jmenovatel je normalizační faktor a musí platit $m > 1$

- V dalším kroku je třeba zjistit nové středy shluků, což se vypočte jako těžiště prvků shluku vážených m-tou mocninou jejich příslušnosti.

$$c_k = \frac{\sum_n \alpha_{n,k}^m x_n}{\sum_n \alpha_{n,k}^m}$$

- Kroky 2 a 3 se budou opakovat tak dlouho, dokud se budou objekty přesouvat mezi shluky nebo se budou středy shluků výrazně měnit.

- Někdy je třeba konečný výsledek ještě defuzzifikovat (jednoznačně přiřadit každý objekt jednomu shluku). To se provede přiřazením objektu do shluku, ke kterému má největší příslušnost.

Příslušnost objektů ve fuzzy shlukování se může vyjádřit běžným zobrazením pomocí množin (viz. obrázek 9) nebo pomocí sloupcových grafů, kde každý objekt bude mít n sloupců (podle počtu shluků) a pomocí nich bude vyjádřena příslušnost k daným shlukům [16] [17].

4.4 Další metody

Metod pro shlukování existuje celá řada a jsou založeny na nejrůznějších principech. Využívají neuronových sítí, teorie grafů, teorie pravděpodobnosti (analýza modů) a další. Není však možné zde uvést všechny tyto metody, a proto alespoň okrajově popíši pár z těchto méně obvyklých metod.

- **Konceptuální shlukování** - cílem je sestrojit klasifikační schéma. To znamená, že se snaží vytvořit jakýsi charakteristický popis pro jednotlivé shluky. Využívá pro to klasifikační strom, s jehož pomocí rozhoduje o přiřazení do shluků [20].

- **Metody neuronových sítí** - jsou to samoorganizující se neuronové sítě (učení bez učitele). Vstupní data jsou postupně předávána síti. Pokud je právě síť v procesu učení, podle hodnot vstupních dat mění svou strukturu. Vstupní data jsou postupně předkládána síti, síť během procesu učení podle jejich hodnot mění svou strukturu. Po předložení vzoru na vstup sítě je vzor zvážen váhovým vektorem každého neuronu. Neuron s nejlepším výsledkem (neboli vítězný neuron) upraví své váhy tak, aby se co nejvíc shodovaly s předloženými daty na vstupu. Všechny neurony si pamatují nastavení svých vah a také informace o svých sousedech (okolní neurony) [21].

- **Bagged clustering** - také nazývaný "Bootstrap aggregating clustering" je poměrně mladou hybridní metodou, která kombinuje hierarchické a nehierarchické metody shlukování. Cílem je zpřesnit (stabilizovat) rozklad množiny dat získaných např. použitím metody *k-průměrů*. Jelikož metoda *k-průměrů* dokáže nalézt jen lokální maxima, je na data uspořádaná touto metodou, znovu aplikována aglomerativní hierarchická metoda. Tím dojde ke stabilizaci výsledku a nalezení globálních maxim [19].

5 Implementace mean-shift

Pro implementaci byl zvolen algoritmus mean-shift. Je zde použit jako segmentační algoritmus tvořící shluky podle podobnosti a blízkosti k největšímu lokálnímu minimu. Obecné informace o algoritmu viz. 4.2. K naprogramování tohoto algoritmu byla použita knihovna *OpenCV*, ze které byly využity metody pro načítání a ukládání obrazu a metody pro přístup k jednotlivým pixelům.

5.1 Použité nástroje a prostředky

Při implemetaci byl použit programovací jazyk C. Patří k nízko úrovňovým jazykům s minimálními nároky na systém a snadnou přenositelností na jiné architektury. Díky těmto vlastnostem se často používá pro systémové programování (ovladače, jádro OS). Jeho vývoj probíhal v letech 1969-1973 v Bellových laboratořích AT&T. Když dosáhl jazyk C poměrně stabilní podoby, byla většina jádra UNIXU přepsána z assembleru do C [24].

Díky své rychlosti, nenáročnosti na HW a lehké přenositelnosti, se používá i pro programování algoritmů pracujících s počítačovou grafikou. Proto byl zvolen jako výchozí jazyk pro tuto implementaci. Pokud bychom však chtěli dosáhnou ještě většího výkonu, lze převést program napsaný v C na architekturu CUDA. Ta slouží k přenesení výpočtu na GPU (procesor grafické karty).

Jako vývojové prostředí jsem použil program Code::Blocks v8.02 s GNU GCC kompilátorem. Pro práci s grafickými prvky, knihovnu OpenCV (kapitola 5.2).

5.2 OpenCV

Tato knihovna byla vyvinuta pro počítačové vidění společností Intel. Později se z ní stala volně šiřitelná knihovna, ale s omezením pro komerční využití (BSD licence). Byla psána pomocí *C*, *C++* a je multiplatformní.

Její vývoj začal v roce 1999 firmou Intel. Jejich cílem bylo vytvořit knihovnu pro počítačové vidění (v reálném čase), která by měla dobře čitelný kód, nezatěžovala by procesor a mohla být snadno využita i v komerčních projektech. Iniciativa pro vytvoření byla také do značné míry ovlivněna tím, že by díky této knihovně mohla stoupnout poptávka po výkonnějších procesorech. Zájem Intelu pomalu ochabl a na vývoji začali pracovat lidé mimo tuto společnost. Po vypuštění více jádrových procesorů se však o vývoj na této knihovně opět začal více zajímat Intel [22].

Knihovna byla portována na téměř každý komerční systém (všetně PowerPC, Macyrobopsi). V tabulce 1 můžeme vidět, na kterých platformách a architekturách byla knihovna zkoušena. Jsou zde uvedeny jen nejběžnější varianty a pokud zde nějaká chybí, neznamená to, že na ní knihovna nepoběží. Knihovna je rozdělena do pěti částí [22]:

- **VC** - základní zpracování obrazu a vyšší úroveň algoritmů počítačového vidění.
- **ML** - strojové učení, statické třídění, seskupovací nástroje.
- **HighGUI** - funkce pro načítání videa, obrázků a práci I/O.

	IA32	EM64T	IA64	Ostatní(PPC, Sparce)
Windows	OK (w.IPP; MSVC6, .NET2005 +OMP, ICC, GCC, BCC)	OK (w.IPP; MSVC6 +PSDK.NET 2005 +OMP, PSDK)	Drobné nedostatky (w.IPP; PSDK)	N/A
Linux	OK (w.IPP; GCC, BCC)	OK (w.IPP; GCC, BCC)	OK (GCC, ICC)	Nefunguje
MacOSX	OK (w.IPP; GCC, nativně APLs)	Netestováno	N/A	OK (iMac G5, GCC, nativně APIs)
Ostatní(BSD, Solaris, . . .)	Nefunguje	Nefunguje	Nefunguje	Hlášena funkčnost na UltraSparc Solaris

Tabulka 1: Platformy na nichž byla testována knihovna OpenCV [22]

- **CXCore** - základní datové struktury.
- **CVAux** - rozpoznávání obličeje HMM, experimentální algoritmy.

5.3 Mean-shift

Jak jsme si již řekli (viz. kapitola 4.2) mean-shift je neparametrická klastrovací metoda, která nevyžaduje zadání počtu shluků. Řadíme ji do skupiny algoritmů *"učení bez učitelé"* (nevyžaduje trénovací data). Nemá problémy s jakýmikoli tvary shluků (např. některé algoritmy se omezují pouze na kruhové oblasti).

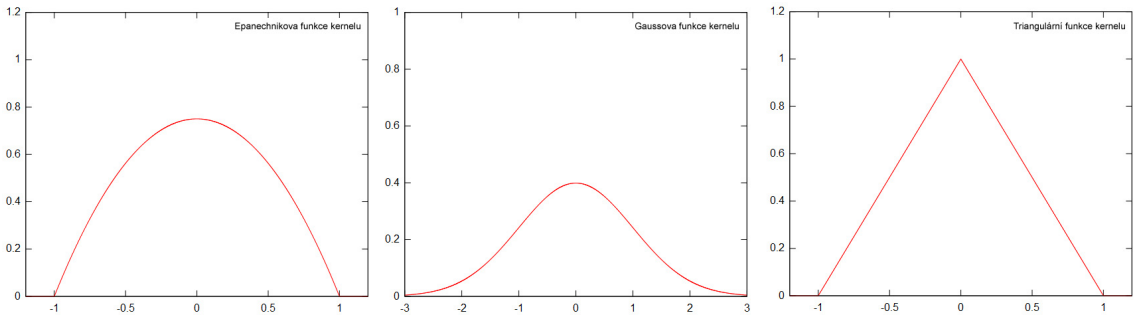
Celý algoritmus bychom si mohli rozdělit do dvou kroků:

- Nalezení lokálního maxima pro všechny pixely
- Seskupení pixelů do jednotlivých shluků na základě vypočtených dat

5.3.1 Popis algoritmu

V tomto prvním a hlavním kroku algoritmu budeme zjišťovat odhad hustoty objektů v rovině. Tento výpočet se provede v okolí každého objektu a následně provedeme posun do bodu s největší hustotou. Toto se bude iterativně opakovat tak dlouho, dokud nedorazíme do bodu, ze kterého se již nelze nikam posunout (nacházíme se tedy v lokálním maximu). Pak již jen zaznamenáme výchozí bod a konečnou pozici [13] [14] [15].

Při implementaci jsem algoritmus aplikoval na obrázek s třemi barevnými složkami RGB, jehož pixely představují všechny shlukované objekty $x_i = (x_i, y_i), i = 1, \ldots, n$ souřadnice pixelů popředí. Při implementaci jsem zjistil, že je dobré si načtený obrázek



Obrázek 10: Funkce profilů kernelu [26]

překonvertovat tak, aby jednotlivé barevné složky dávaly hodnoty z intervalu $\langle 0, 1\rangle$, což nám později usnadní některé výpočty. Odhad hustoty rozložení objektů(pixelů) v rovině získáme dle předpisu [13] [14] [15]

$$\tilde{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

kde n je celkový počet pixelů, d pak označuje dimenzi prostoru (pro šedotónní obraz to je 2, pro RGB 5), x zde zastupuje bod vůči kterému je počítána hustota (většinou střed oblasti zájmu), h je poloměr okna(oblasti zájmu). Také je potřebné uvést si definici kernelu $K(x)$.

$$K(x) = c_{k,d} k\left(\|x\|^2\right)$$

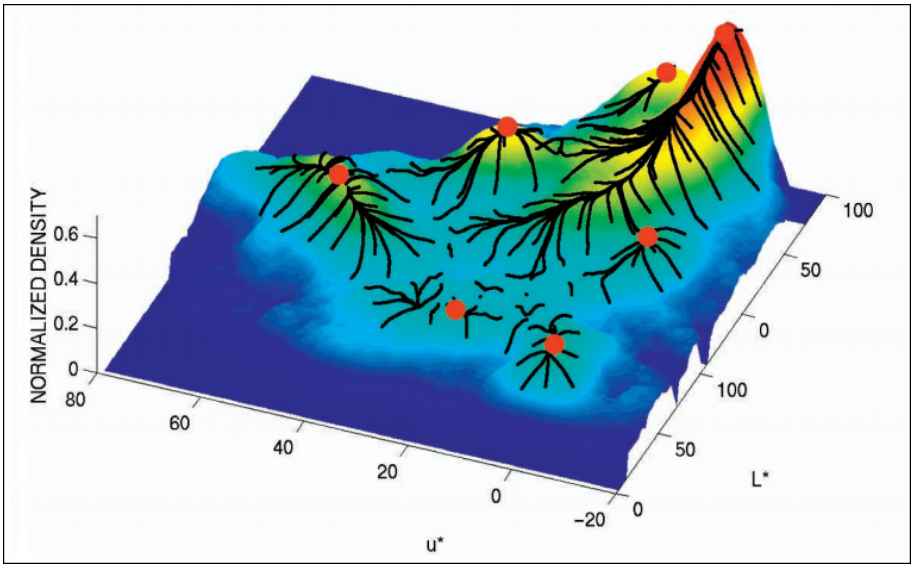
Parametr $c_{k,d}k$ je normalizační konstanta, která závisí na volbě funkce $k(x)$ označované jako profil kernelu K. K čemu vlastně profil kernelu slouží? Pomocí profilu kernelu si můžeme definovat tvar oblasti zájmu, což je jakési *"okénko"* pomocí nějž budeme vyhledávat lokální maximum (viz. obrázek 8 - čárkovaná křivka). Při implementaci jsem si zvolil tzv. *Epanechnikův kernel*. Pro algoritmus mean-shift se používá vcelku často, definice je jednoduchá a tvarem bude nejvíce připomínat kruhovou oblast. Definice epanechnikova kernelu [13] [14] [15]:

$$k(z) = \begin{cases} 1 - z & \text{pro} \quad z \leq 1 \\ 0 & \text{jinak} \end{cases} \tag{1}$$

Abychom si vůbec udělali představu, jak mohou vypadat další funkce kernelu, podívejme se na obrázek 10, kde můžeme vidět epanechnikovu, gaussovu a triangulační funkci kernelu (popisováno zleva).

Nyní jsme schopni si vypočítat odhad hustoty rozložení bodů v rovině, to nám však nestačí. Potřebujeme získat lokální maxima hustoty $\tilde{f}$, ležící v nulových bodech jejího gradientu. Odhad gradientu můžeme získat z již vypočtené funkce díky tomu, že platí:

$$\tilde{\nabla} f_{h,K}(x) \equiv \nabla \tilde{f}_{h,K}(x) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x - x_i) k'\left(\left\|\frac{x - x_i}{h}\right\|^2\right)$$



Obrázek 11: Postup mean-shiftu zobrazený v 3D modelu [13]

Nyní je třeba si definovat funkci $g(x)$, která nám upraví funkci profilu kernelu do patřičného tvaru

$$g\left(x\right) = -k'\left(x\right).$$

Po této úpravě můžeme dále pokračovat ve výpočtu gradientu hustoty

$$\begin{aligned}
\nabla \tilde{f}_{h,K}\left(x\right) &= \frac{2c_{k,d}}{nh_{d+2}} \sum_{i=1}^{n} \left(x_i - x\right) g\left(\left\| \frac{x - x_i}{h} \right\|^2\right) \\
&= \frac{2c_{k,d}}{nh_{d+2}} \left[\sum_{i=1}^{n} g\left(\left\| \frac{x - x_i}{h} \right\|^2\right)\right] \left[\underbrace{\frac{\sum_{i=1}^{n} x_i g\left(\left\| \frac{x-x_i}{h} \right\|^2\right)}{\sum_{i=1}^{n} x_i \left(\left\| \frac{x-x_i}{h} \right\|^2\right)}}_{m(c)} - x\right]
\end{aligned}$$

Z naposledy uvedeného vzorce je velmi důležitá část $m(x)$, která vypočítá tzv. mean-shift vektor, směřující vždy do místa s největším nárůstem odhadnuté hustoty $\tilde{f}$. Pomocí iterativního volání tak můžeme dostat trajektorii bodu x_i, vedoucí do lokálního maxima. Opakovaný výpočet mean-shift vektoru by se měl ukončit právě tehdy, když $\nabla \tilde{f}_{h,K}\left(x\right)$ bude rovna nule. V praxi se však místo toho volí nějaké dostatečně malé číslo. Výpočetní čas i výkon bude menší a výsledek téměř totožný. Nyní, po vypočtení všech trajektorií bodů x_i, přiřadíme každému lokálnímu maximu nějakou barvu (náhodně či podle klíče) a každý bod obrazu obarvíme podle lokálního maxima, ve kterém končí jeho trajektorie. Tím je provedena celá segmentace. Trajektorie bodů vedoucích k lokálnímu maximu můžeme vidět na 3D modelu v obrázku 11 [13] [14] [15].

Pro lepší pochopení bude dobré celý algoritmus ještě jednou ve zkratce popsat.

- Pro každý bod obrazu budeme postupně počítat vektor posuvu ($x(x)$), což opakovaně aplikujeme i na tento vektor, do té doby než gradient hustoty $\nabla \tilde{f}_{h,K}(x)$ bude roven nule nebo jinému dostatečně malému číslu.
- Každému lokálnímu maximu přiřadíme barevnou hodnotu, ať už náhodně nebo podle předem zvolených parametrů. Každý bod obrazu bude obarven podle lokálního maxima, do kterého *"došel"* pomocí mean-shiftu.

6 Testování

Následně po vytvoření algoritmu bylo zapotřebí provést řadu testů, zda se algoritmus chová tak, jak by měl. Prvním testem bylo zkontrolování posuvů do lokálního maxima. Postačil k tomu jednoduchý obrázek s gradientem (postupný růst sytosti barvy). Mean-shift se pomocí přesunů dostal až do místa s největší sytostí barvy, což je správné chování. Dalším krokem bylo spuštění algoritmu na již normální obraz a vypozorování zda dochází ke shlukování, což se projeví vytvořením větších ploch se stejnou barvou. Tyto plochy nám říkají, že všechny body s jednou barvou *"došly"* pomocí mean-shiftu do stejného lokálního maxima. Jako poslední bylo potřeba zjistit, jak se algoritmus chová při změně hodnoty h (poloměru okna mean-shiftu). Teoreticky by se měl celkový počet segmentů zvětšovat, pokud budeme hodnotu h snižovat, jelikož mean-shift bude mít menší *"rozhled* pro nalezení místa s největší hustotou. Proto by mělo docházet k častějšímu nalezení lokálního maxima a tím i vytvoření segmentu.

Postupně jsem provedl segmentaci s různou velikostí parametru $h \in \{5, 10, 20, 30, 40\}$. Předpoklad se splnil a počet segmentů klesal s rostoucí velikostí h. Doba zpracování však byla mnohem delší, což je zapříčiněno tím, že při každem výpočtu posunu mean-shift se muselo počítat s čím dál větším okolím středového bodu. Volba parametru je tedy velmi důležitá, pokud bychom si zvolili příliš malou hodnotu, vytvoří se příliš mnoho shluků a tím se ztíží i následující zpracování obrazu. Pokud však, ale zvolíme příliš velkou hodnotu, může dojít k splynutí důležitých shluků a opět nám to znemožní další analýzu. Na obrázku 12 a 13 můžeme vidět postup testování segmentace se změnou parametru h. Je patrné, že druhá část obrázku obsahující 936 segmentů je méně vhodná k další analýze, než třeba čtvrtá či pátá část (obsahující 42 segmentů).

Obrázek 12: Test segmetace: původní obrázek, parametr h = 5, parametr h = 10

Obrázek 13: Test segmetace: parametr h = 20, parametr h = 30, parametr h = 40

7 Závěr

Během vypracovávání jsem zjistil, že metod segmentace je velké množství a metod shlukování taktéž. Díky tomu jsou shlukovací metody téměř v každém zdroji děleny jiným způsobem, a proto jsem zvolil rozdělení do větších celků, které jsou zmíněny téměř všude. Z těchto skupin jsem pak vybral typické segmentačních algoritmy a popsal jejich princip.

Další částí byla implementace jednoho z algoritmů. Zvolil jsem si shlukovací metodu mean-shift a začal se jí zabývat. Během studia tohoto algoritmu mě také zaujalo jeho využití při sledování objektů v *real-time* videu. Algoritmus mean-shift je poměrně jednoduchý, ale podle mě má příliš velké výpočetní nároky. Což by se samozřejmě dalo velmi zredukovat pokud by se implementovaný algoritmus upravil na vícevláknovou aplikaci a nebo přepracoval pro zpracování GPU (grafický procesor), pomocí technologie CUDA. Výsledná segmentace provedená tímto algoritmem je však velmi dobrá, samozřejmě za předpokladu, že uživatel zvolí rozumnou velikost *okna* mean-shift.

Radomír Hladík

8 Literatura

[1] LINKA, Aleš. E-Learning - Segmentace obrazu [online].
Dostupné z: http://e-learning.tul.cz/ [cit. 25.3.2010].

[2] STRAKA, Stanislav. Segmentace obrazu - Úvod. Brno 2009. 3 s. Diplomová práce na fakultě Informatiky Masarykovy Univerzity. Vedoucí diplomové práce Mgr.Radka Pospíšiilová

[3] WALES, Jimmy; SANGER, Larry. Wikipedia - Segmentation (image processing) [online].
Dostupné z: http://www.wikipedia.org/ [cit. 1.4.2010].

[4] STRAKA, Stanislav. Segmentace obrazu - Metody sedmentace. Brno 2009. 4-24 s. Diplomová práce na fakultě Informatiky Masarykovy Univerzity. Vedoucí diplomové práce Mgr.Radka Pospíšiilová

[5] STRAKA, Stanislav. Segmentace obrazu, obrázek č. 2.7 quadtree pro split and merge. Brno 2009. 19 s. Diplomová práce na fakultě Informatiky Masarykovy Univerzity. Vedoucí diplomové práce Mgr.Radka Pospíšiilová

[6] SOJKA, Eduard. Digitální zpracování a anylýza obrazu - segmentace obrazu. Ostrava 2000. 74-92 s. ISBN ISBN 80-7078-746-5. Skripta na fakultě elektro techniky a informatiky Vysoké školy Báňské.

[7] ŽÁK, Libor . Shluková analýza I. Automatizace, březen 2004, roč. 47, č. 3, s. 180-182

[8] KELBEL, Jan; ŠILHÁN, David. Shluková analýza. 2-4 s. Odborný článek získaný z Českého vysokého učení technického v Praze.

[9] WALES, Jimmy; SANGER, Larry. Wikipedia - k-means clustering [online].
Dostupné z: http://www.wikipedia.org/ [cit. 12.4.2010].

[10] ŽÁK, Libor . Shluková analýza II. Automatizace, duben 2004, roč. 47, č. 4, s. 251-253

[11] WALES, Jimmy; SANGER, Larry. Wikipedia - Cluster analysis [online].
Dostupné z: http://www.wikipedia.org/ [cit. 15.4.2010].

[12] WALES, Jimmy; SANGER, Larry. Wikipedia - k-means clustering, obrázek č.1 [online].
Dostupné z: http://www.wikipedia.org/ [cit. 15.4.2010].

[13] COMANICIU, Dorin; MEER, Peter . Mean Shift: A Robust Approach Toward Feature Space Analysis. Transactions on pattern analysis and machine intelligence, květen 2002, roč. 24, č. 5, s. 603-619

[14] WALES, Jimmy; SANGER, Larry. Wikipedia - Mean-shift [online].
Dostupné z: http://www.wikipedia.org/ [cit. 18.4.2010].

[15] DOUBEK, Petr. Mean-shift segmentace. Praha 2007. 1-4 s. Odborný článek získaný z Českého vysokého učení technického v Praze.

[16] ŽÁK, Libor . Shluková analýza III. Automatizace, květen 2004, roč. 47, č 5, s. 320-322

[17] Dipartimento di Elettronica ed informazione - Fuzzy C-Means Clustering [online]. Dostupné z: http://home.dei.polimi.it/ [cit. 22.4.2010].

[18] ŽÁK, Libor . Shluková analýza III. Automatizace, obrázek č.9, květen 2004, roč. 47, č 5, s. 322

[19] FISCHER, Bernd; BUHMANN, Joachim . Mean Shift: A Robust Approach Toward Feature Space Analysis. Transactions on pattern analysis and machine intelligence, květen 2003, roč. 25, č. 11, s. 1-6

[20] DVOŘÁK, Jiří. Expertní systémy. Brno 2004. 56-57 s. Učební texty z Fakultě Informatiky Vysoké učení technické v Brně

[21] HLAVÁČ, Václav. Expertní systémy. Brno 2009. Učební texty z Centrum strojového vnímání Katedra kybernetiky FEL Českého vysokého učení technického v Praze.

[22] LAPÁČEK, Ondřej. Využití knihovny OpenCV pro zpracování obrazu v mobilních robotických systémech. Zlín 2009. 13-16 s. Bakalářská práce na Fakultě aplikované informatiky Univerzity Tomáše Bati. Vedoucí bakalářské práce Erik Král.

[23] ŠPANĚL, Michal. Obrazové segmentační techniky. Brno 2005. 24-25 s. Studijní dokumentace pro fakultu informačních technologií, Vysoké učení technické v Brně.

[24] WALES, Jimmy; SANGER, Larry. Wikipedia - C (programovací jazyk)[online]. Dostupné z: http://www.wikipedia.org/ [cit. 30.4.2010].

[25] GERPANIS, Konstantinos . Mean Shift Clustering, obrázek č.1, srpen 2005, s. 3

[26] WALES, Jimmy; SANGER, Larry. Wikipedia - Kernel (statistics), obrázek č.2,3,5 [online].
Dostupné z: http://www.wikipedia.org/ [cit. 1.5.2010].